CS3530: Computer Networks - I

SOCKET PROGRAMMING PROJECT

Professor: Kotaro Kataoka

Multimedia Group Chat:

Team Members:

SHAIK MOHAMMED SAYEED - CS19BTECH11004

GNANESWAR BVS - CS19BTECH11007

GANTASALA NAGA ANEESH AJAROY - CS19BTECH11010

NAGA HARI TEJA PEDDI - CS19BTECH11021

VEMULAPALLI ADITYA - CS19BTECH11025

BEDARAKOTA AMOGH - CS19BTECH11031

Github Link: https://github.com/mohsayeed/Multi-Media-Group-Chat

TABLE OF CONTENTS:

PROJECT DESCRIPTION

This project is a multimedia group chat application which is built upon the socket programming completely from scratch using send/recv functions. It has some enhanced features along with simple group chat among the clients like multi-file transfer, end-to-end encryption(e2e), Emojis can be used during texting, Video calling for one-on-one and simple GUI.

PROTOCOL USED

With UDP and TCP as options for our group chat , we chose TCP because it gives us guaranteed transfer of data by using a proper established connection unlike the UDP protocol where data loss might happen.

INBUILT FEATURES OF PYTHON USED

send : The send() function initiates transmission of a message from the specified socket to another socket.

recv : The recv() function shall receive a message from a connection-mode or connectionless-mode socket.

gethostname : gethostname() function is used to get the hostname of a computer. This function returns a string containing the hostname of the machine where the Python interpreter is currently executing.

connect : The connect() function attempts to make a connection on a socket. This functions takes the 3 arguments; *socket* , *address* , *address_len*

threads : A multi-threaded server application can accept a client connection, create a thread for that conversation, and continue listening for requests from additional clients using threads. A thread is a sequence of instructions which run independently of the program and of any other threads

Modules Used: socket - Used for dealing with sockets,

threading - Used for dealing with threads

tkinter - Used to create a simple graphical user interface

emoji - Used to integrate emojis in the text messages

FILE STRUCTURE

See github for a bit of understanding.

Important Note: Here we took the number of clients in the group to be a maximum of 3 . If we want a group which contains more number of clients , then extra extra folders for each client must be added where the server is present and also number of clients in the server code is set to be 3 which must be changed to desired value, and extra client code files must also be needed.

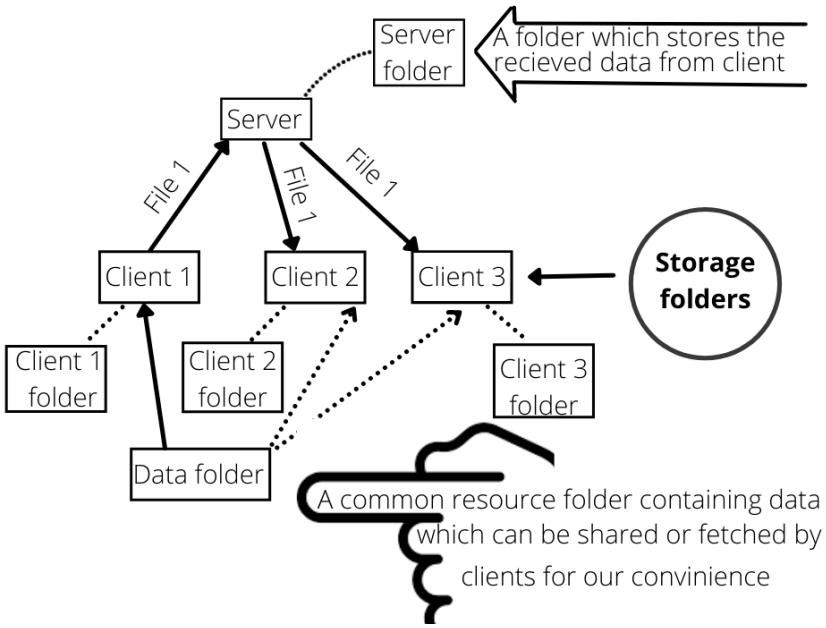
There will be a source folder which contains data (which contains all data files). The clients need to send these data files to other clients. For example client1 writes a command **FILESEND s.txt** which means that client1 has to send s.txt to other clients (client2 and client3).

After this the server detects the above command and understands that a file needs to be sent. It becomes ready to receive the data sent by client (here client1). After the server gets the data it creates a separate folder (with name 'server') and puts the data in that folder.

Once this is done the server sends the data to other clients (client2 and client3). Other clients receive this data and put them in their client2 and client3 folders.

We have 3 separate folders for each client and a separate folder for the server.

Illustration of File 1 transfer from client 1 to other clients via server



COMMANDS TO EXECUTE

We first open our terminal windows (client and server). In each terminal we go to the directory where our python files are present. We give the command 'py filename.py' (if this command doesn't work try 'python3 filename.py'). After we run this command for the server program , in the terminal we can see the IP address of the particular Local Host. When we run the above command for client programs, in the terminal we get an input prompt to 'enter the IP address of the server '. The client can paste the server IP address here to connect to the server. Thus a Connection is established. In the server terminal a message is displayed saying that 'a client has joined'. After that a GUI (Graphical User Interface). Note: If you close the terminal window you get a message saying ' forced closed connection'. If you want to close you can directly use 'QUIT' command or close the command prompt.

CHAT OPTIONS AVAILABLE

There are 4 chat options to execute:

- **FILESEND filename** : When we run this command the particular datafile from the data folder is sent to the server. Server stores the data in a 'server' folder. The datafile from this server folder is sent to other clients
- **MEMBERS ONLINE** : The user who uses this option gets a list of currently active members in the chat room other than the user himself.
- **QUIT** : Connection gets closed between client and server and the client no longer remains in the group and cannot chat. To reconnect, the client needs to start the joining process again.
- If there is a message which is not of above 2 formats then it will be considered as a text message.

PROJECT DESIGN

BASELINE IMPLEMENTATION

- It has group chat features where a server is present and a group of clients can connect to this server. Server creates a socket and binds it with an IP number and port number. Then the listen method enables our server and waits for requests from the client.
- A client must create a socket and then connect to the server socket using the connect method.
- send and recv inbuilt functions are used for communication between server and clients.
- Threading is used for both clients and servers. For the server, threads are used so that it can take care of multiple clients where each thread takes care of a single client. For a client, threading is helpful when it wants to receive and send at the same time, for this we use the main thread and an extra thread which we create.

ENHANCED FEATURES ADDED

SIMPLE GUI

We developed a GUI for this project using a python library called '**tkinter**'. This library gives us some good features such as message list, where if we use message list.insert then messages get inserted which are displayed in the terminal.

For our Project the Chat Window needs to be initialised. This is done by using the tkinter library. In the chat window we can add many custom features. For example, the scrolling option can be changed,and the appearance of the side grid can be changed. For our project we created a Simple chat window without much customisations.

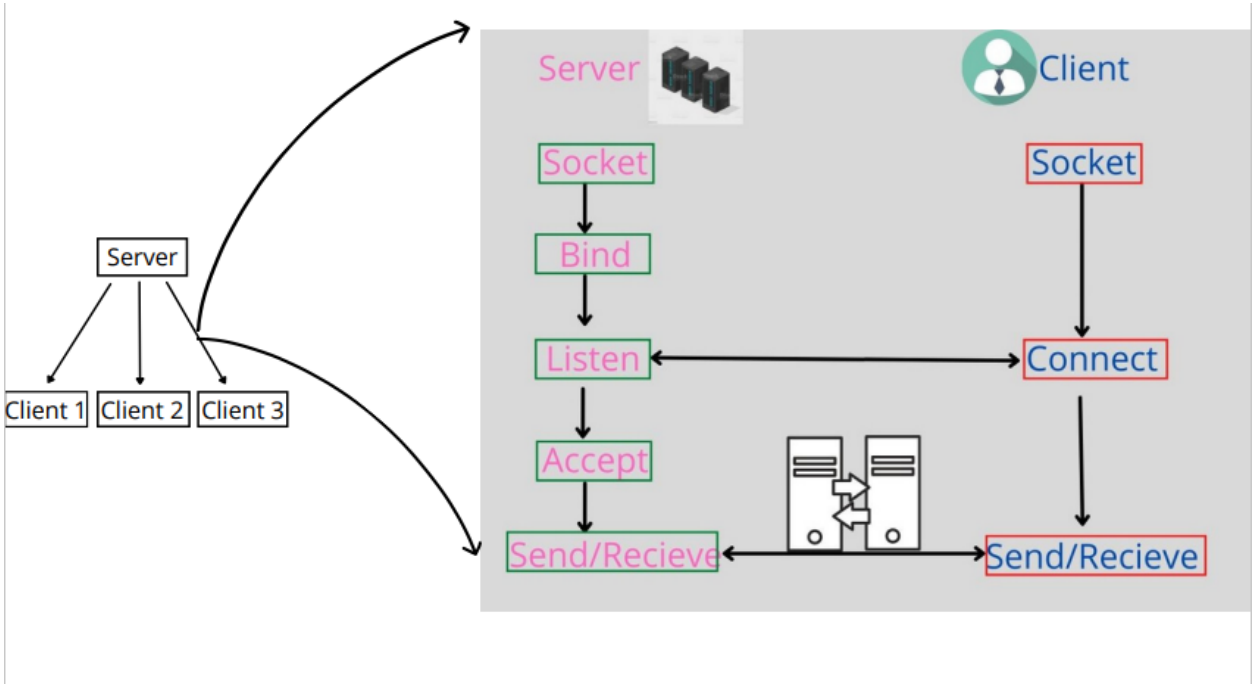
GROUP CHAT

There can be multiple clients connected to the server and they can send messages to each other, sen663.d files in the group.

FILE TRANSFER

The files are divided into chunks before sending and then these chunks will be delivered in sequence for this we use inbuilt functions of send and recv.

FLOW CHART



RUNNING THE CODE

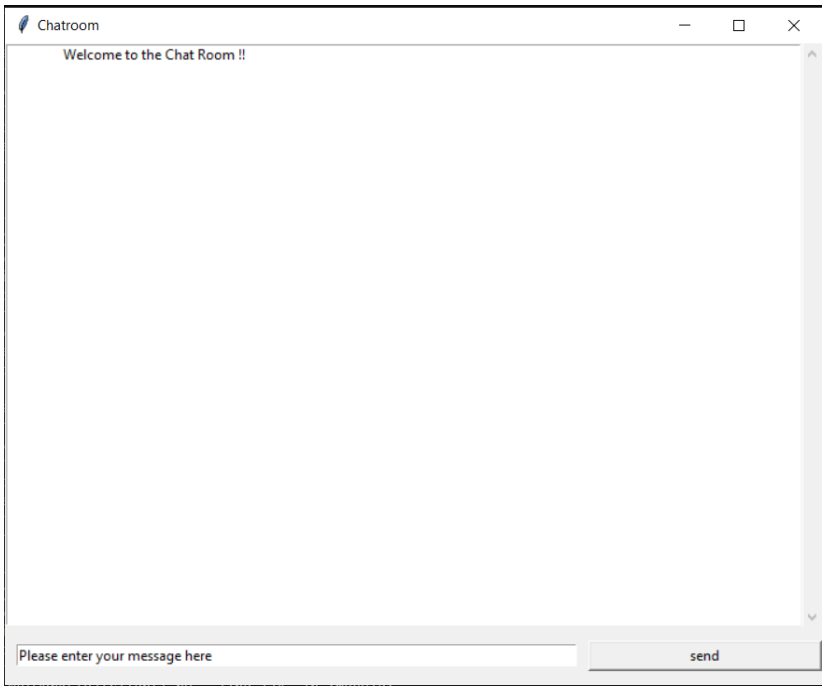
Starting the Server

```
D:\Sem 5\Networks\V3>py server.py
Server IP Address : 192.168.0.103
Server Connected
Hostname of server socket : DESKTOP-OP1TCPB
```

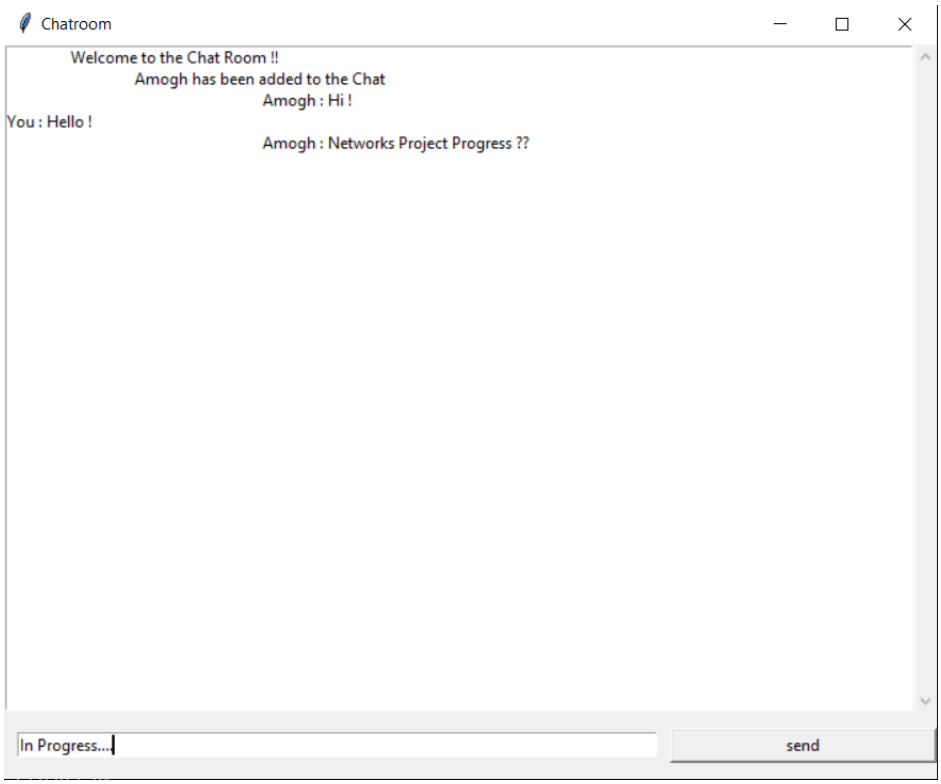
Connecting the client to server

```
D:\Sem 5\Networks\V3>py client1.py
Please Provide the Server IP Address : 192.168.0.103
Connected to Server
Please enter your Name : Hari
Transfering you to Chat Room....
```

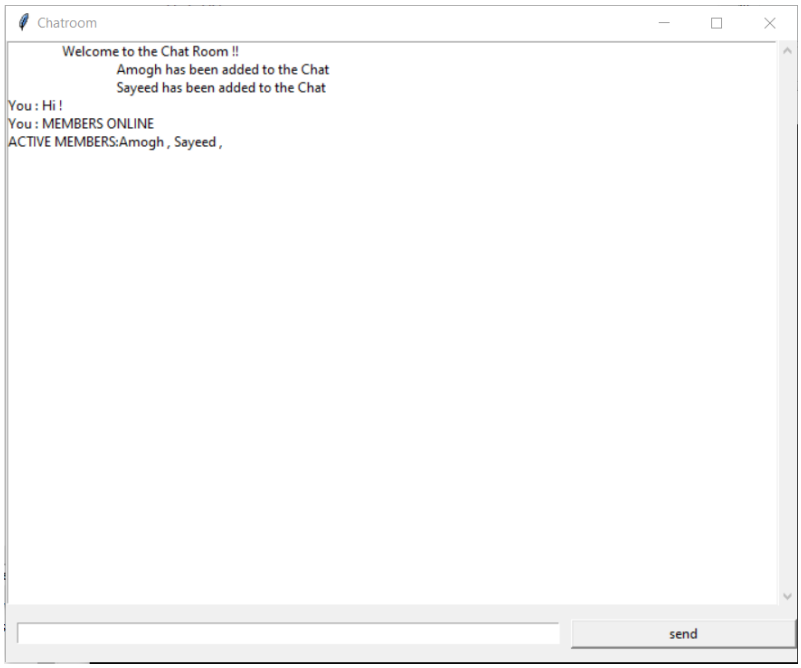
Then a chatbox graphical user interface opens....



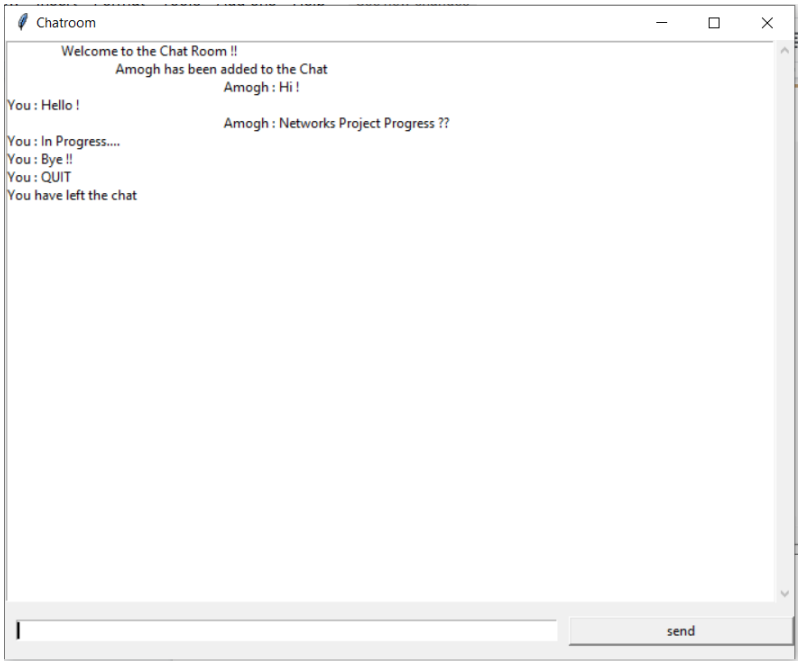
You can start chatting by typing messages in the text box at the bottom of window and click send button to send the message into the chat room



You can get the active members in the chat room by using MEMBERS ONLINE command



You can leave the chat room using the QUIT command and can close the window after leaving the chat room



REFERENCES

https://docs.python.org/3/library/socket.html

https://docs.python.org/3/library/tkinter.html

https://medium.com/analytics-vidhya/how-to-print-emojis-using-python-2e4f93443f7e